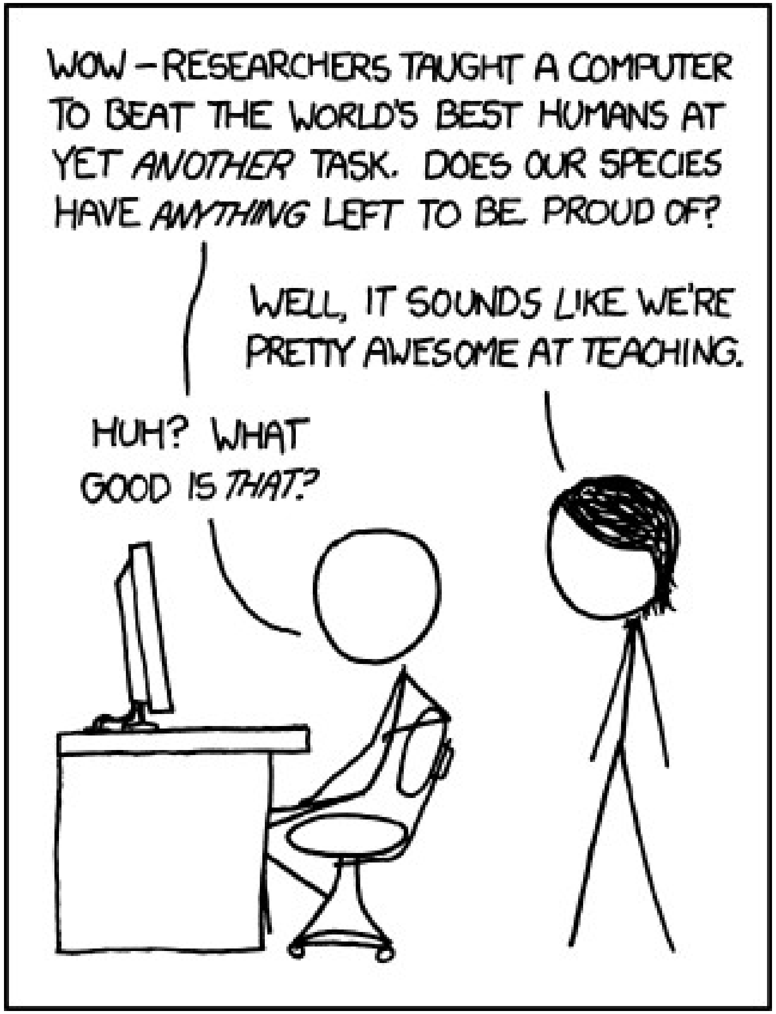


http://xkcd.com/894/

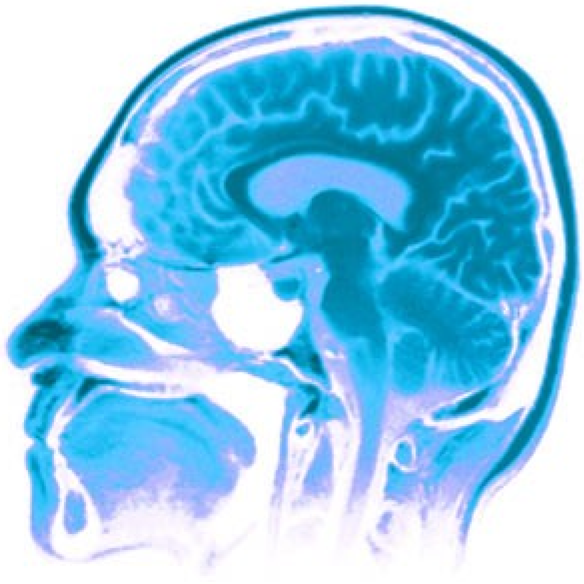
Neural Networks

David Kauchak/
Joseph C. Osborn
CS51A
Fall 2019

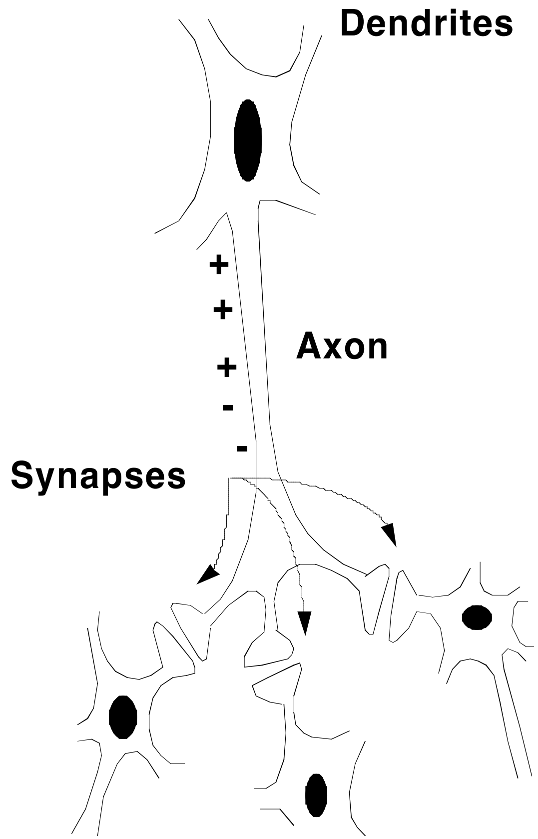
Neural Networks

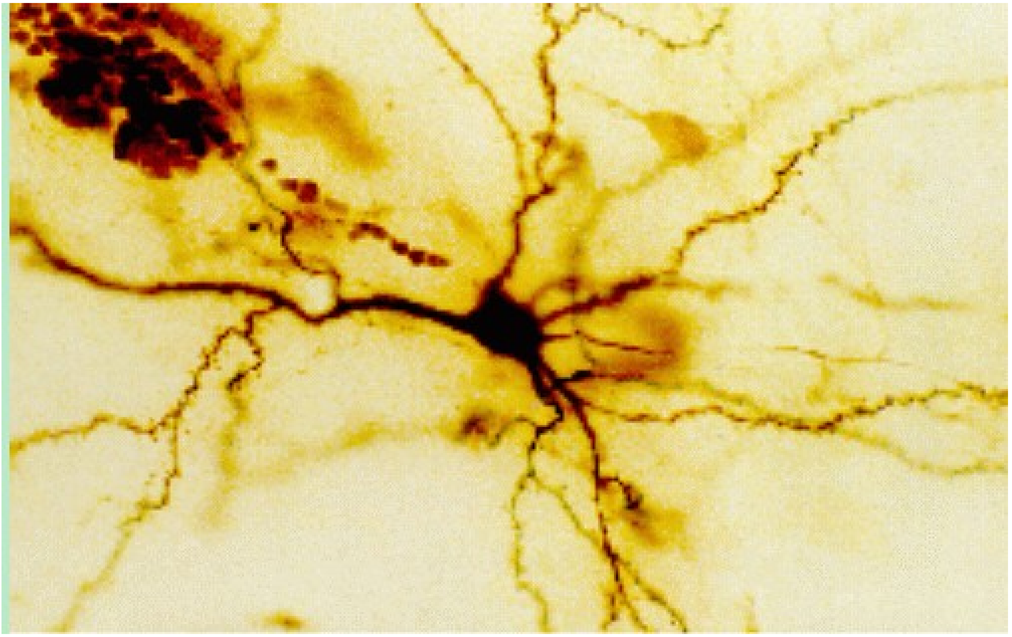
Neural Networks try to mimic the structure and function of our nervous system

People like biologically motivated approaches



Our Nervous System

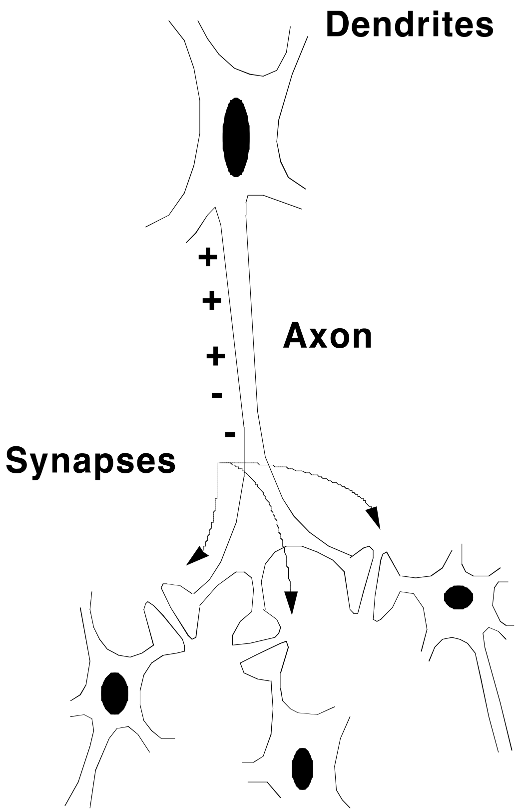




Neuron

What do you know?

Our nervous system:
the computer science view

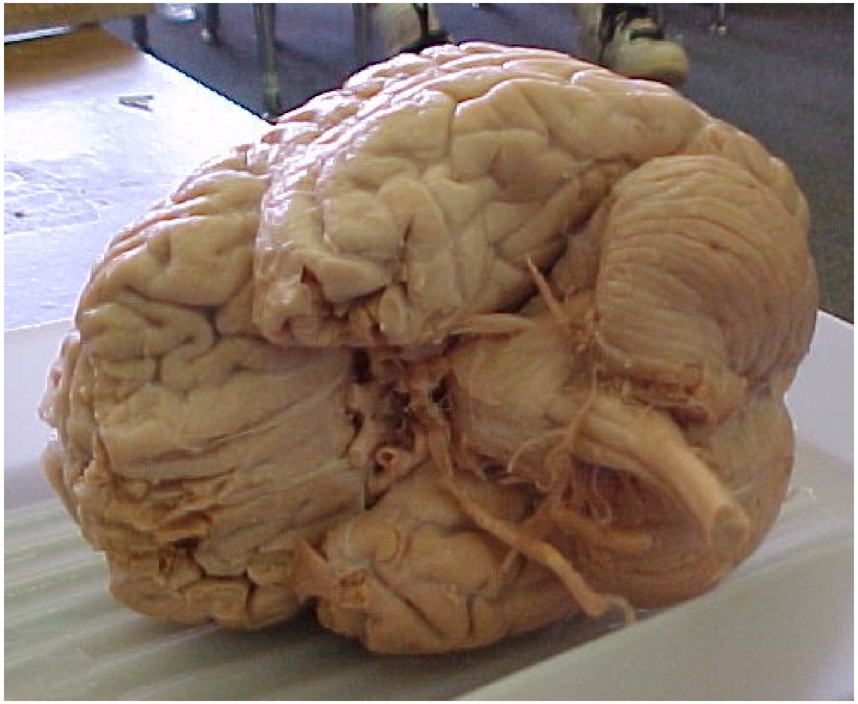


the human brain is a large collection of interconnected neurons

a NEURON is a brain cell

- they collect, process, and disseminate electrical signals
- they are connected via synapses
- they FIRE depending on the conditions of the neighboring neurons

Our nervous system

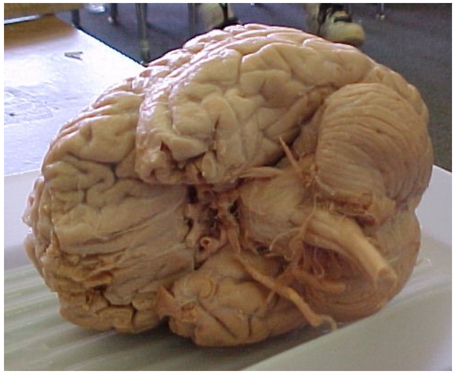


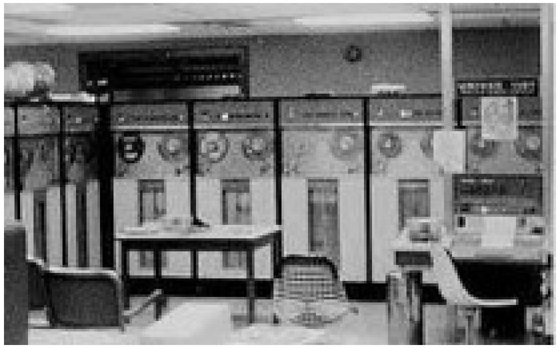
The human brain

- contains ~10^{11} (100 billion) neurons
- each neuron is connected to ~10^4 (10,000) other neurons
- Neurons can fire as fast as 10^{-3} seconds

How does this compare to a computer?

Man vs. Machine





10^{11} neurons	10^{10} transistors
10^{11} neurons	10^{11} bits of ram/memory
10^{14} synapses	10^{13} bits on disk
10^{-3} “cycle” time	10^{-9} cycle time

Brains are still pretty fast

Who is this?

Brains are still pretty fast

If you follow basketball, you'd be able to identify this person in under a second!

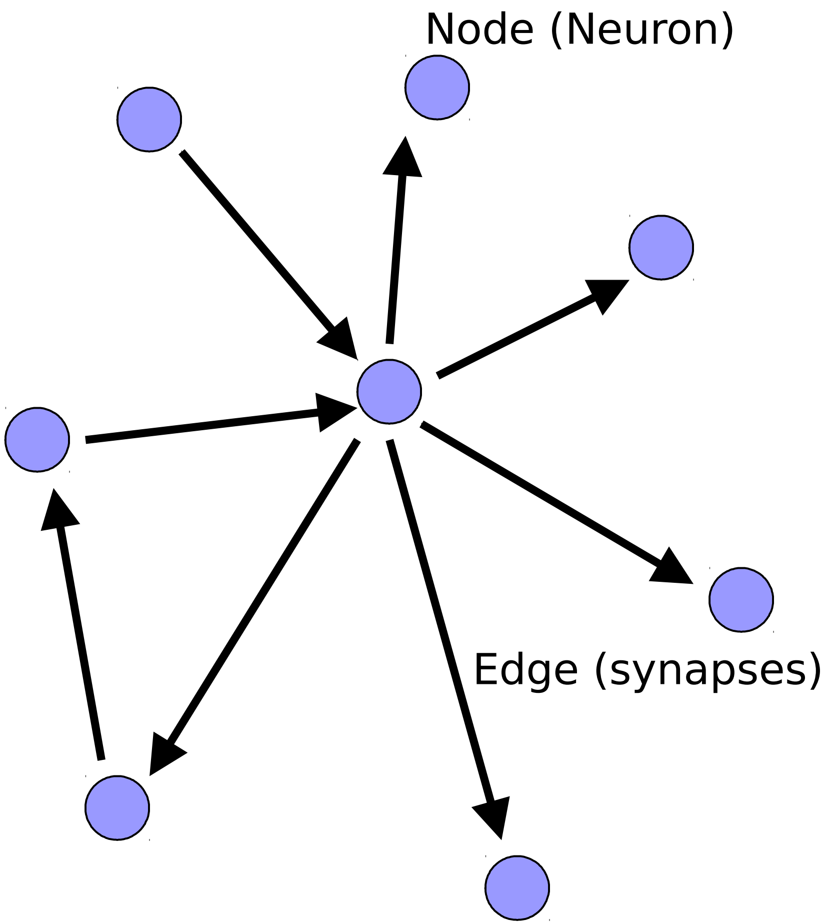
Given a neuron firing time of 10^{-3} s, how many neurons in sequence could fire in this time?

- A few hundred, maybe a thousand

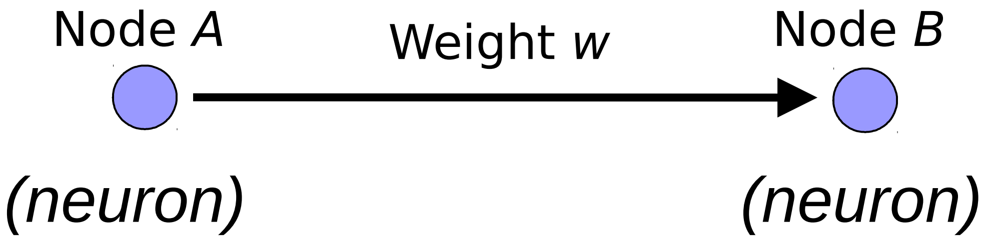
What are possible explanations?

- either neurons are performing some very complicated computations
- brain is taking advantage of the **massive** parallelization (remember, neurons are connected ~10,000 other neurons)

Artificial Neural Networks



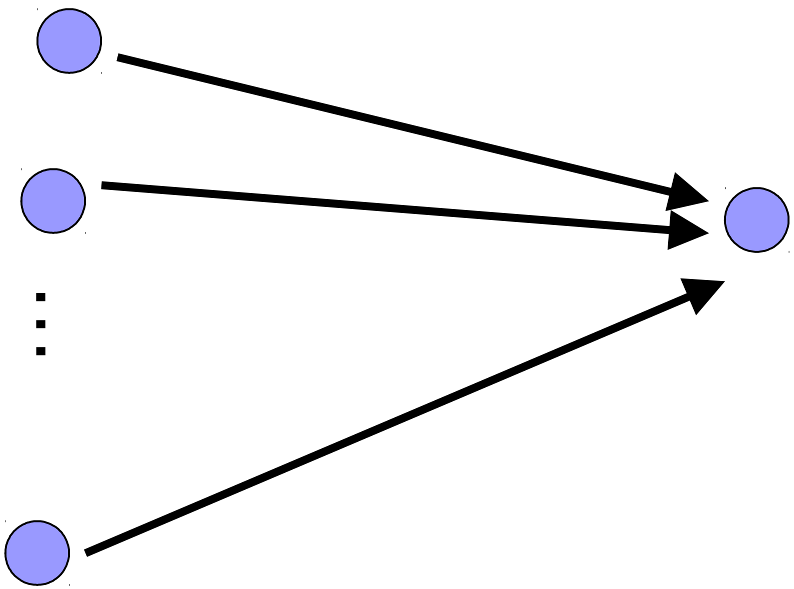
our approximation



W is the strength of signal sent between A and B.

If *A* fires and *w* is **positive**, then *A* **stimulates** *B*.

If *A fires* and *w* is **negative**, then *A* **inhibits** *B*.



A given neuron has many, many connecting, input neurons

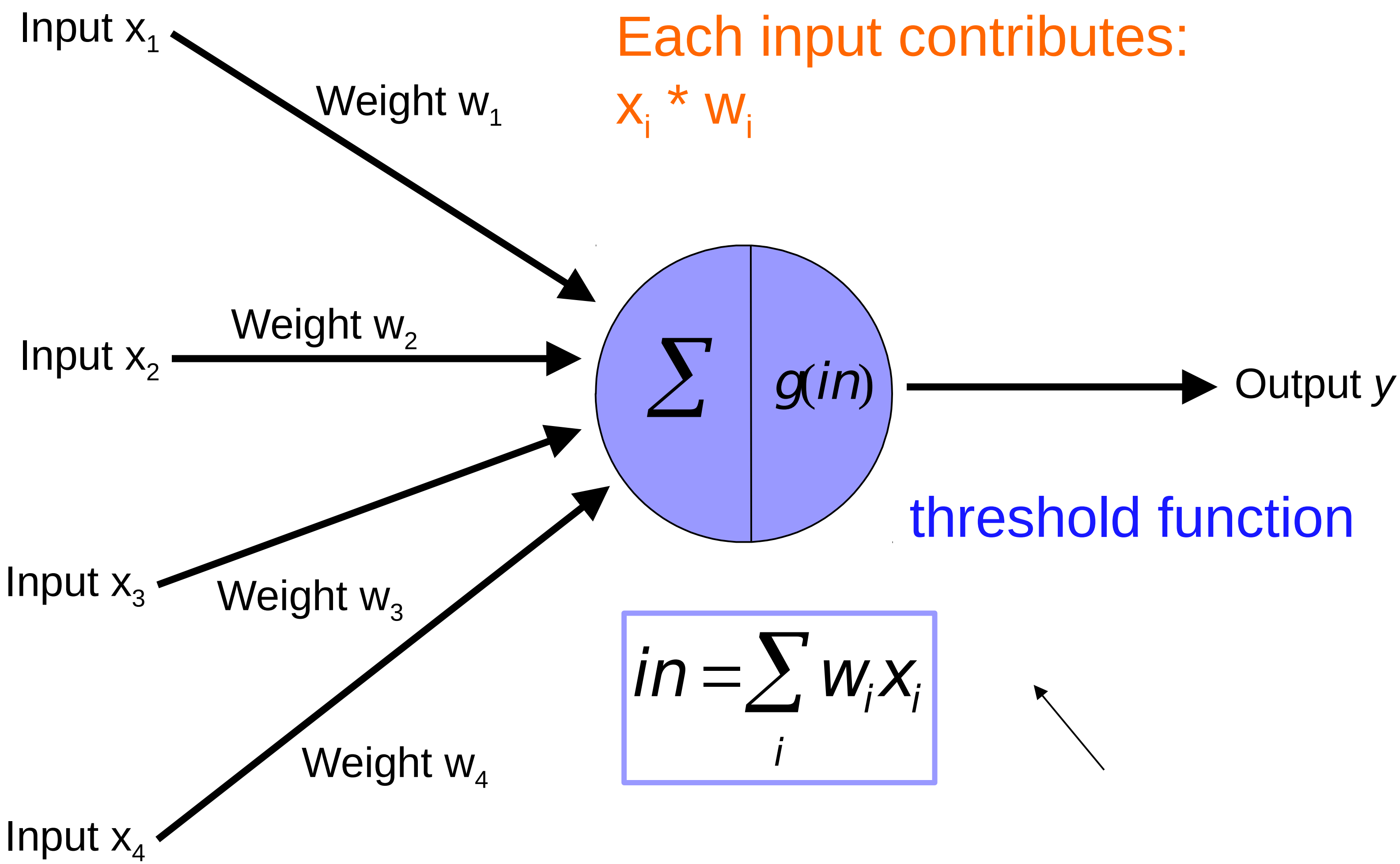
If a neuron is stimulated enough, then it also fires

How much stimulation is required is determined by its **threshold**

A Single Neuron/Perceptron

Each input contributes:

$x_i * w_i$

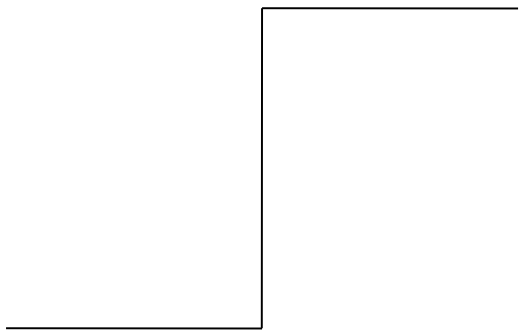


$$\boxed{in = \sum_i w_i x_i}$$

Possible threshold functions

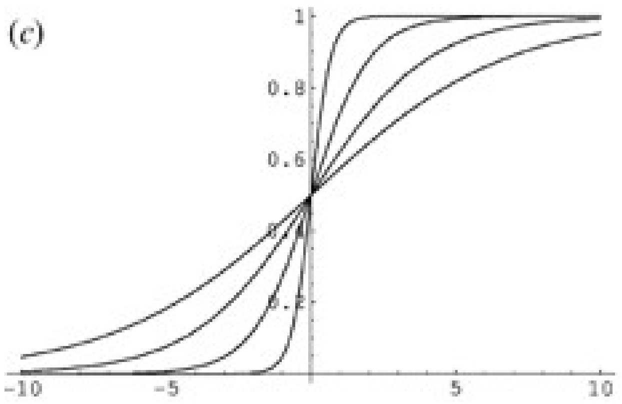
hard threshold

$$g(x) = \begin{cases} 1 & \mathit{if}\ \mathrm{x} \geq \mathit{threshold} \\ 0 & \mathit{otherwise} \end{cases}$$

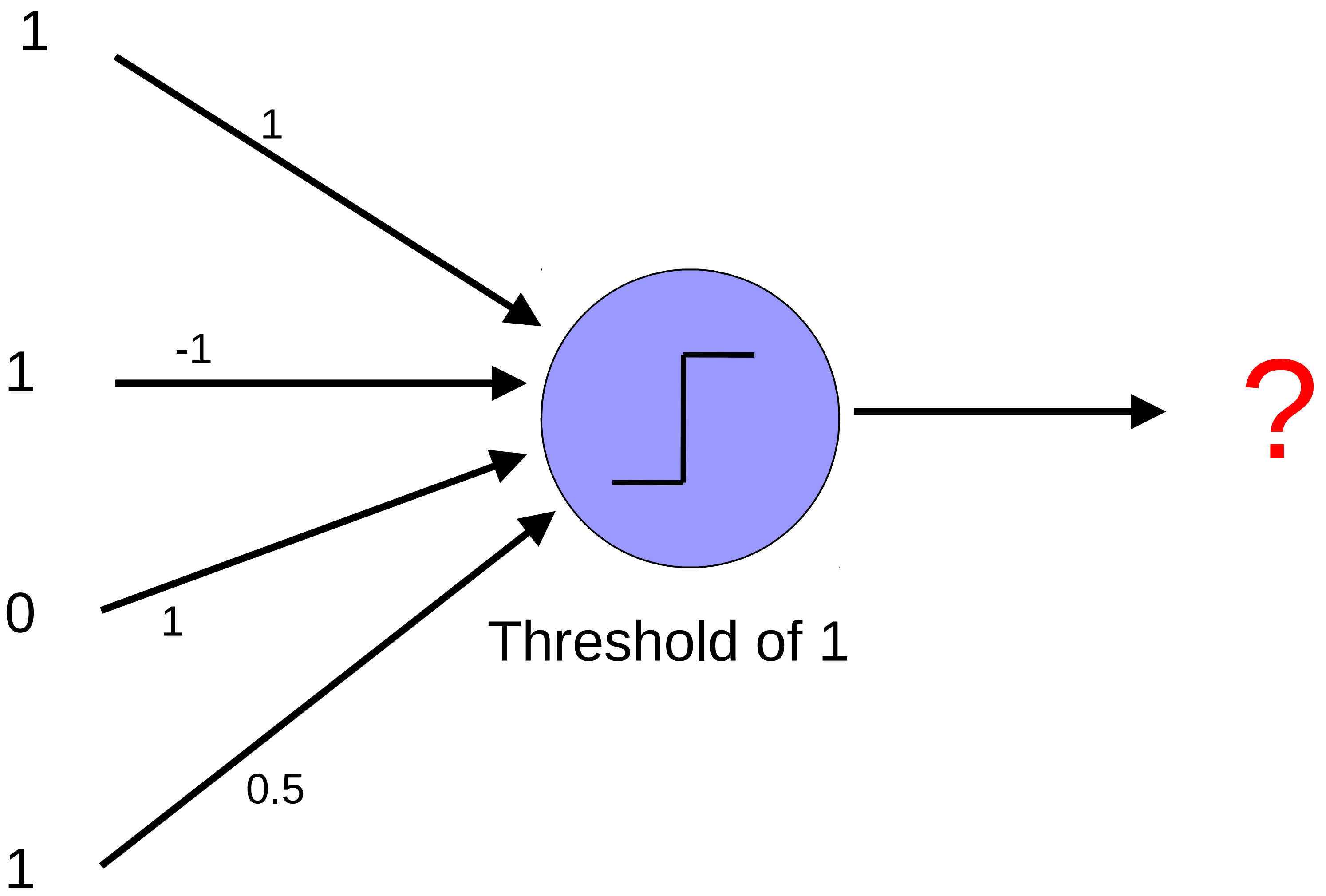


sigmoid

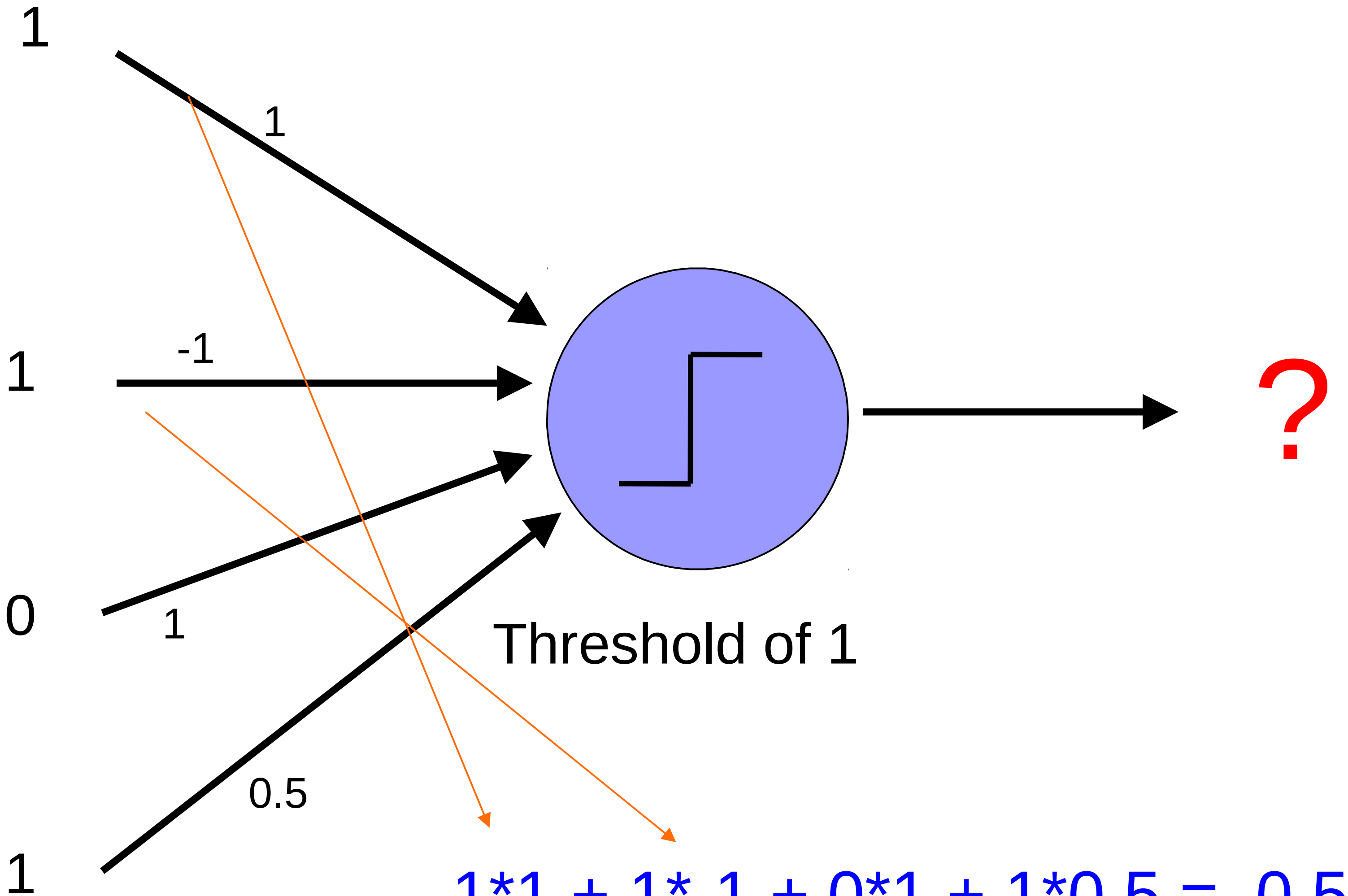
$$g(x) = \frac{1}{1 + e^{-ax}}$$



A Single Neuron/Perceptron

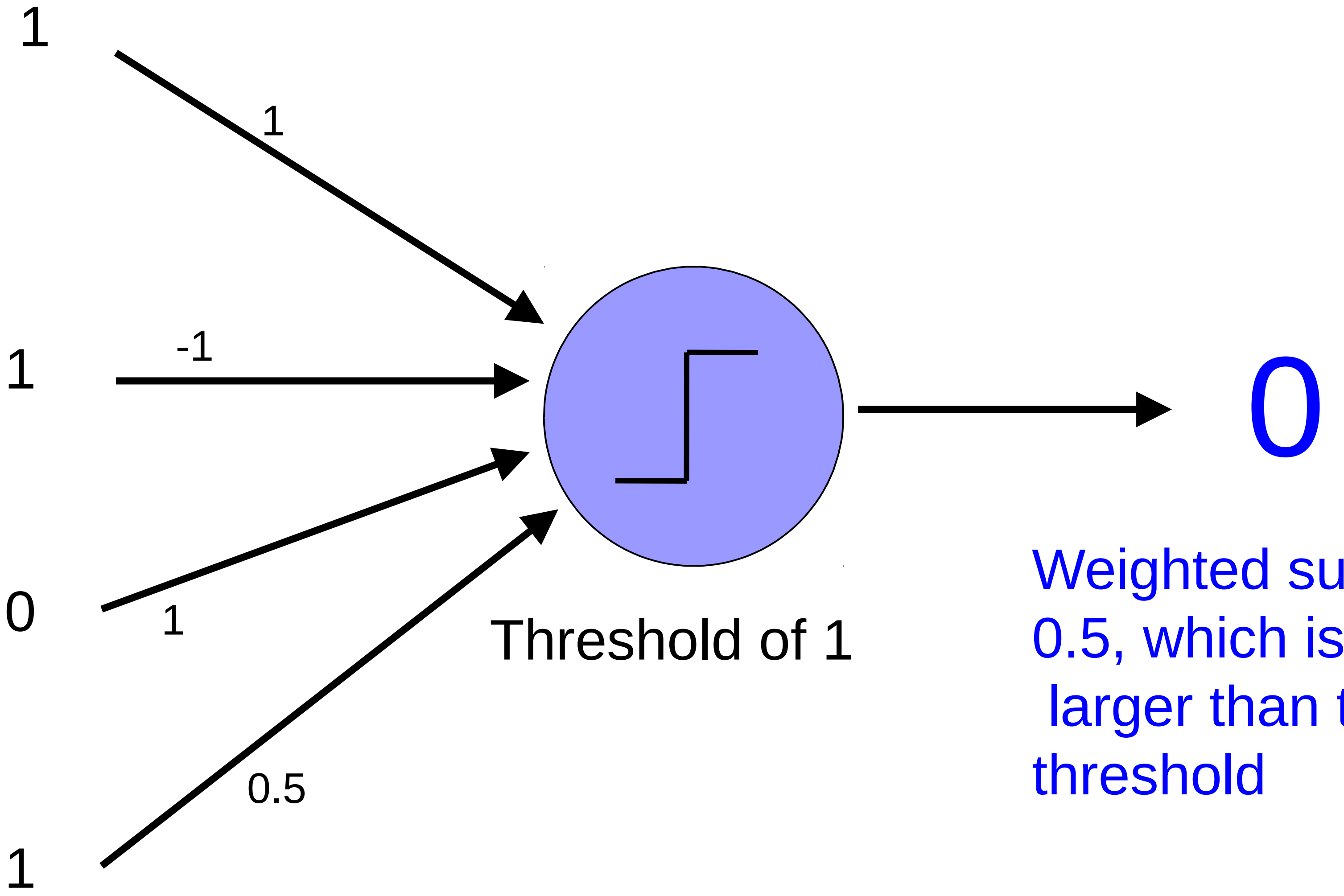


A Single Neuron/Perceptron



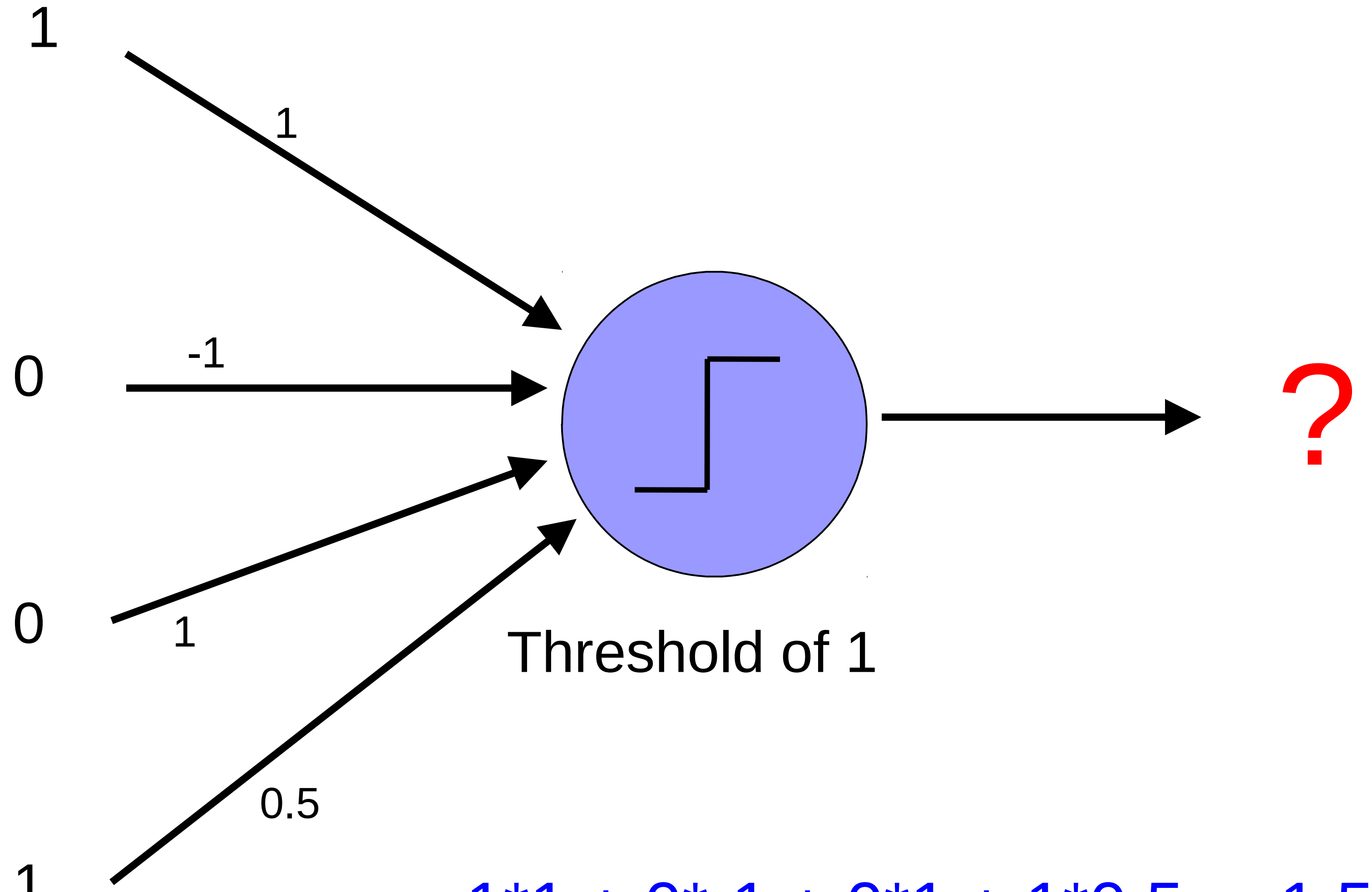
1*1 + 1*-1 + 0*1 + 1*0.5 = 0.5

A Single Neuron/Perceptron



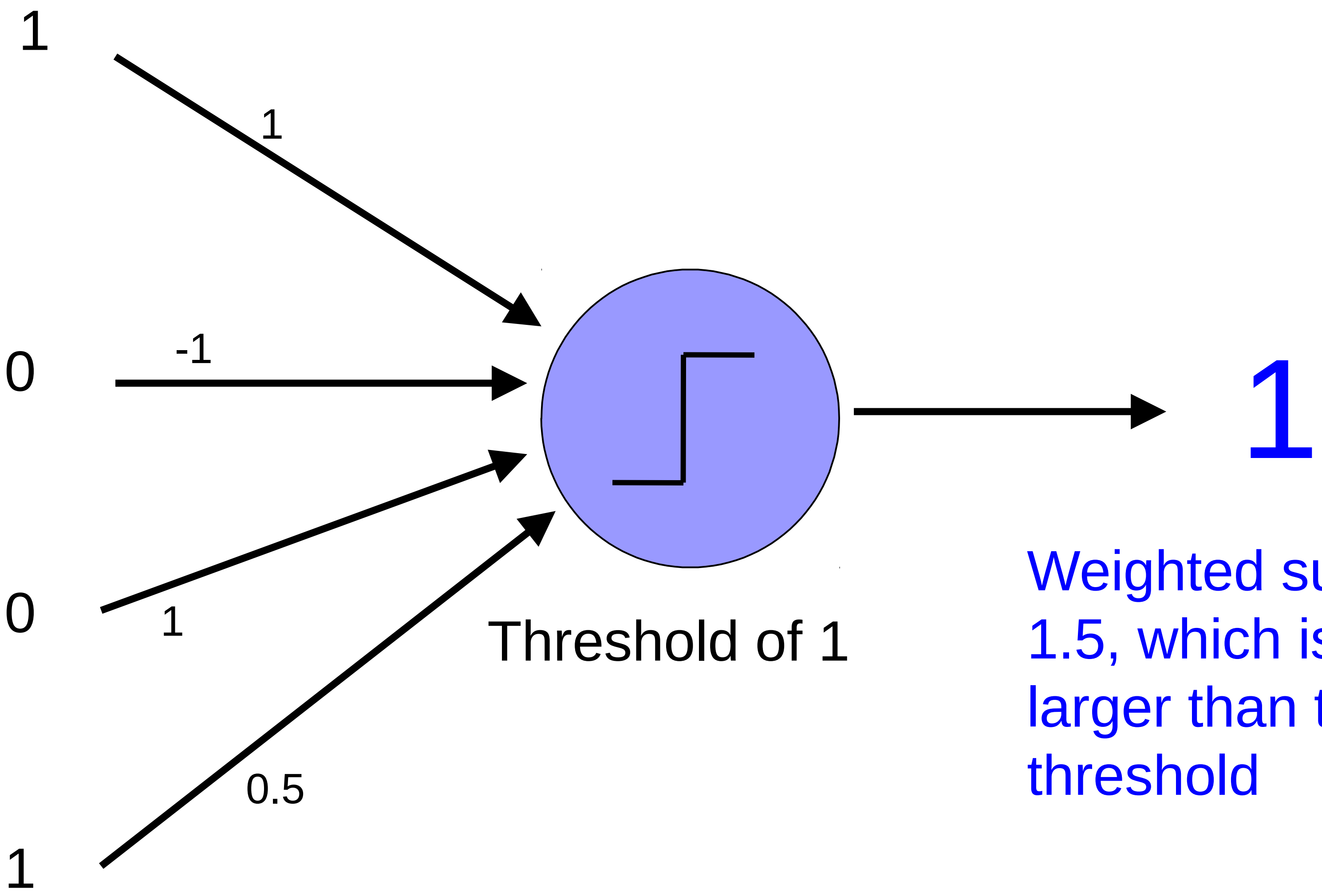
Weighted sum is 0.5, which is not larger than the threshold

A Single Neuron/Perceptron



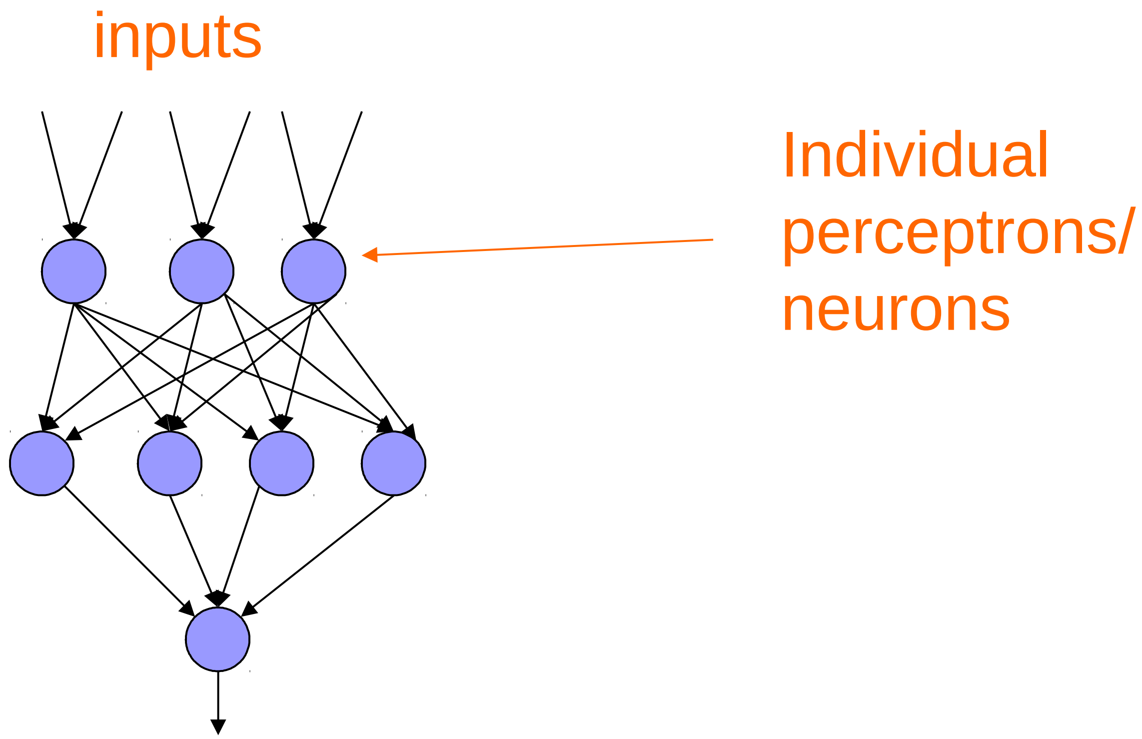
1*1 + 0*-1 + 0*1 + 1*0.5 = 1.5

A Single Neuron/Perceptron

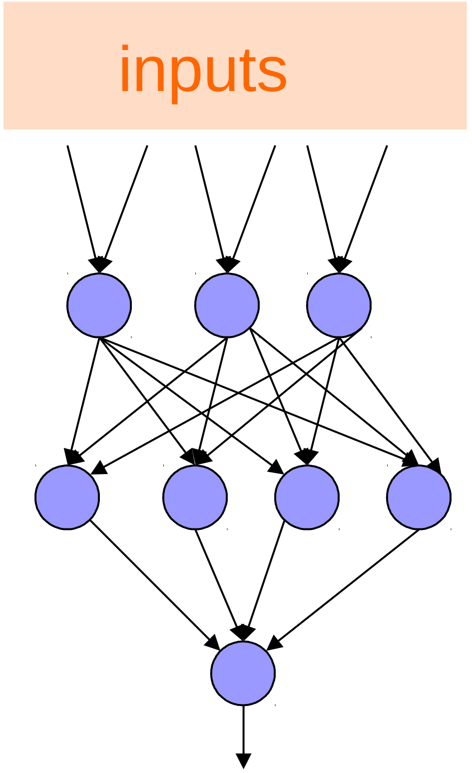


Weighted sum is 1.5, which is larger than the threshold

Neural network



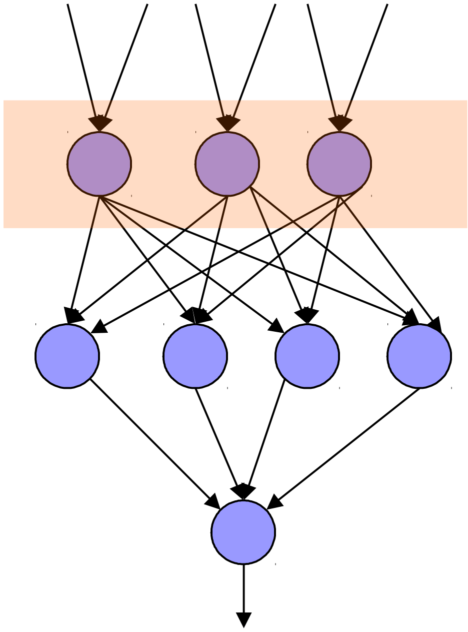
Neural network



some inputs are provided/entered

Neural network

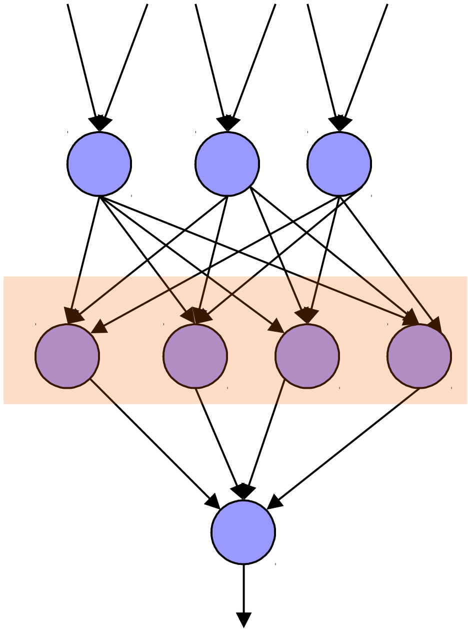
inputs



each perceptron computes and calculates an answer

Neural network

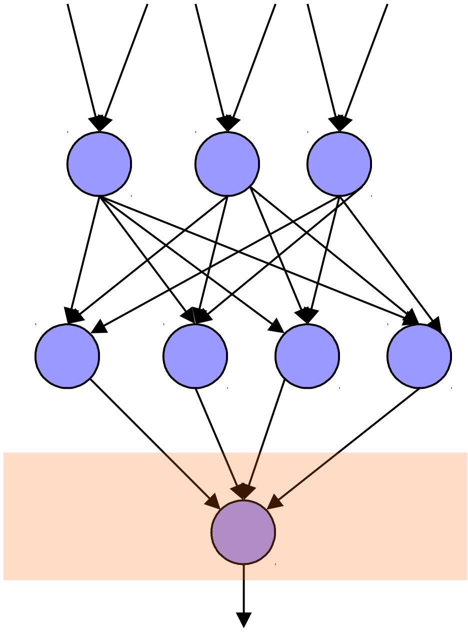
inputs



those answers become inputs for the next level

Neural network

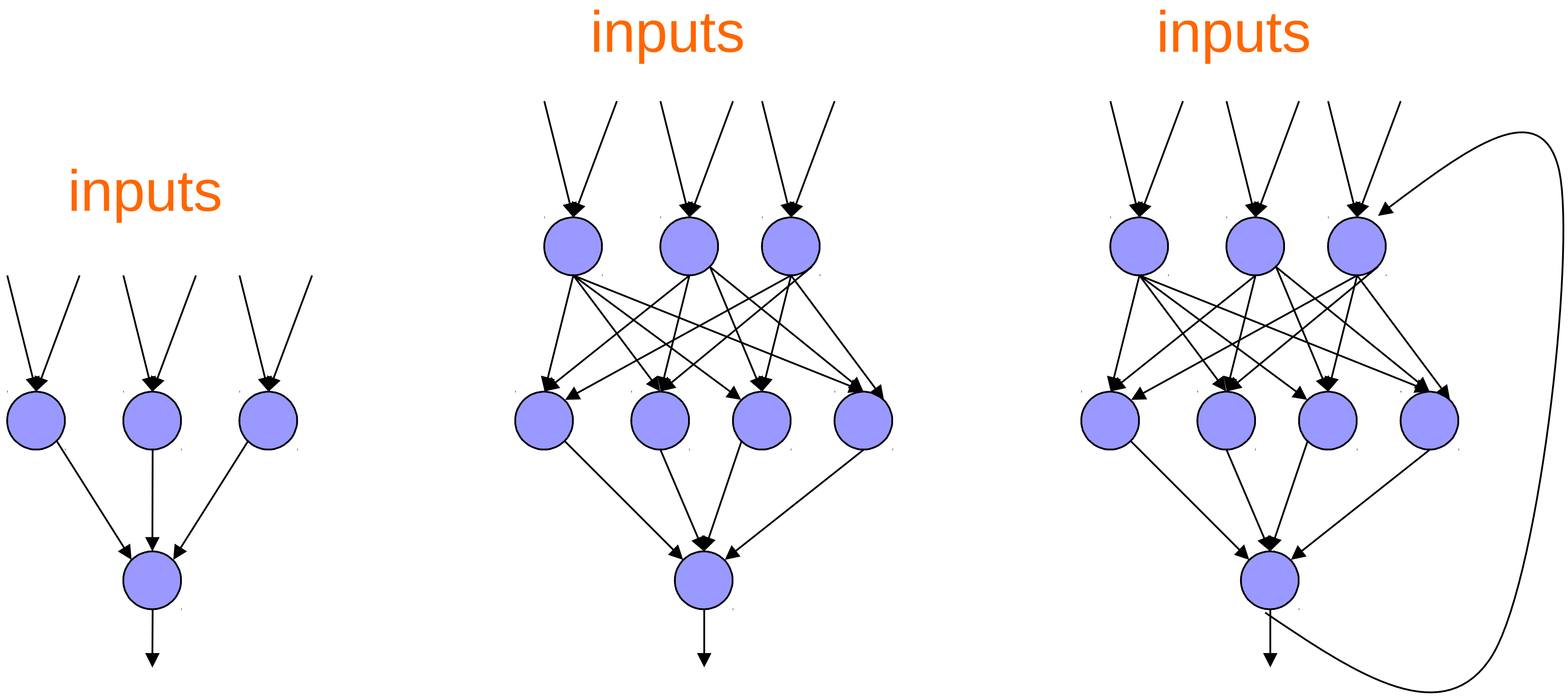
inputs



finally get the answer
after all levels compute

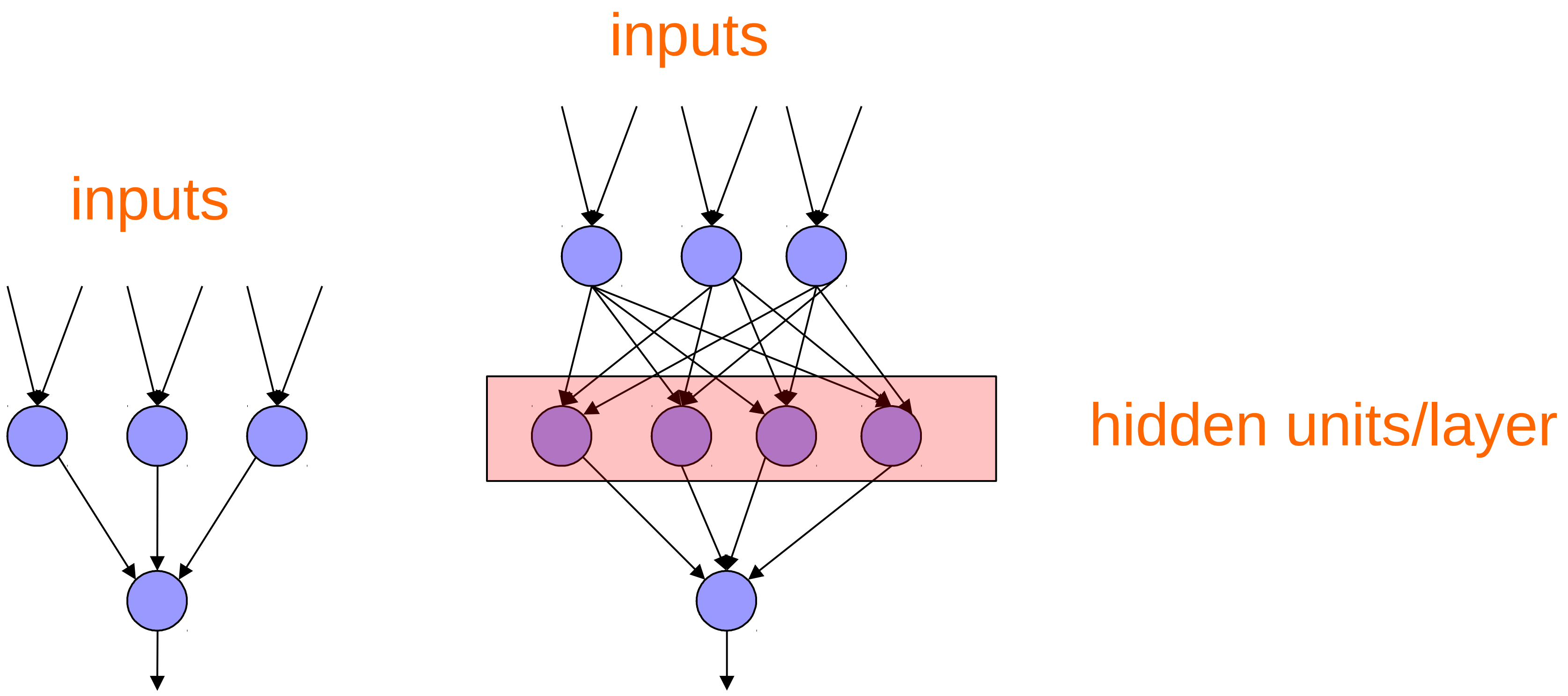
Neural networks

Different kinds/characteristics of networks



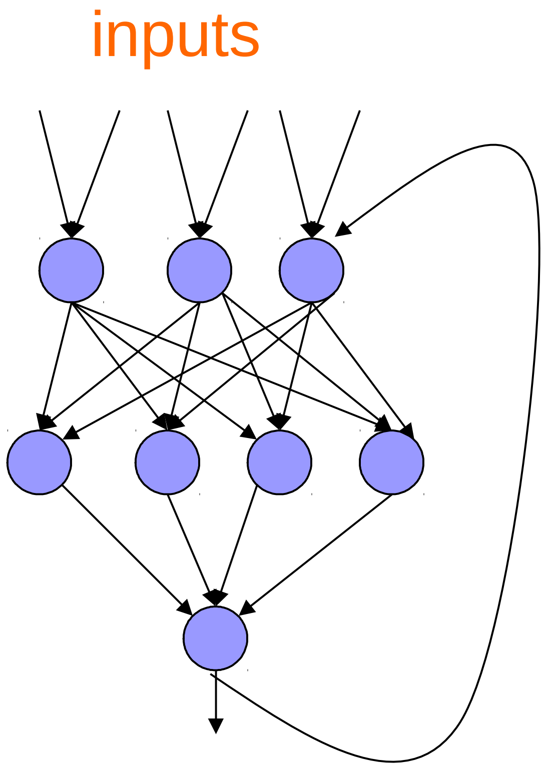
How are these different?

Neural networks



Feed forward networks

Neural networks



Recurrent network

Output is fed back to input

Can support memory!

How?

History of Neural Networks

McCulloch and Pitts (1943) – introduced model of artificial neurons and suggested they could learn

Hebb (1949) – Simple updating rule for learning

Rosenblatt (1962) - the *perceptron* model

Minsky and Papert (1969) – wrote *Perceptrons*

Bryson and Ho (1969, but largely ignored until 1980s--Rosenblatt) – invented back-propagation learning for multilayer networks

Training the perceptron

First wave in neural networks in the 1960’s

Single neuron

Trainable: its threshold and input weights can be modified

If the neuron doesn’t give the desired output, then it has made a mistake

Input weights and threshold can be changed according to a learning algorithm

Examples - Logical operators

AND – if all inputs are 1, return 1, otherwise return 0

OR – if at least one input is 1, return 1, otherwise return 0

NOT – return the opposite of the input

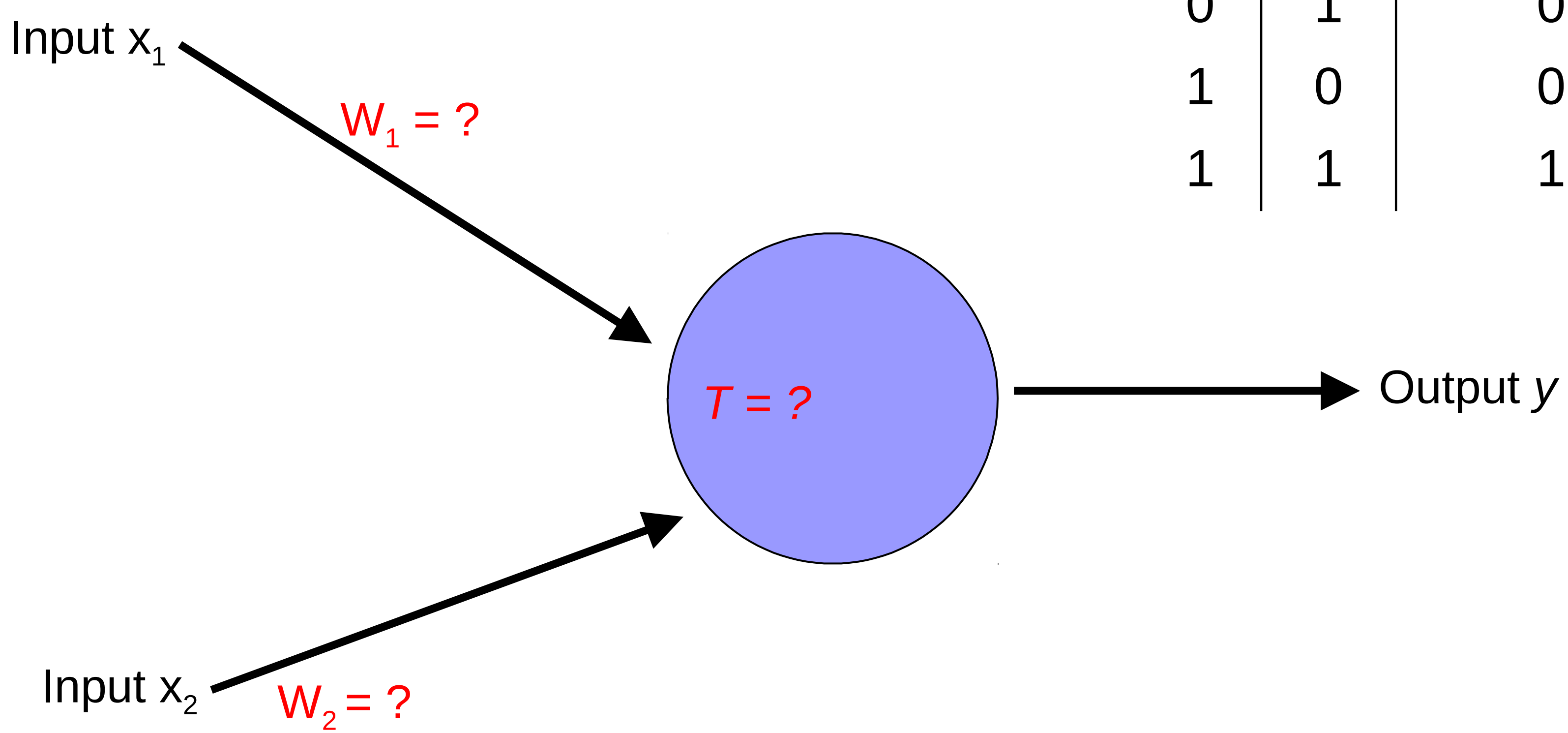
XOR – if exactly one input is 1, then return 1, otherwise return 0

AND

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

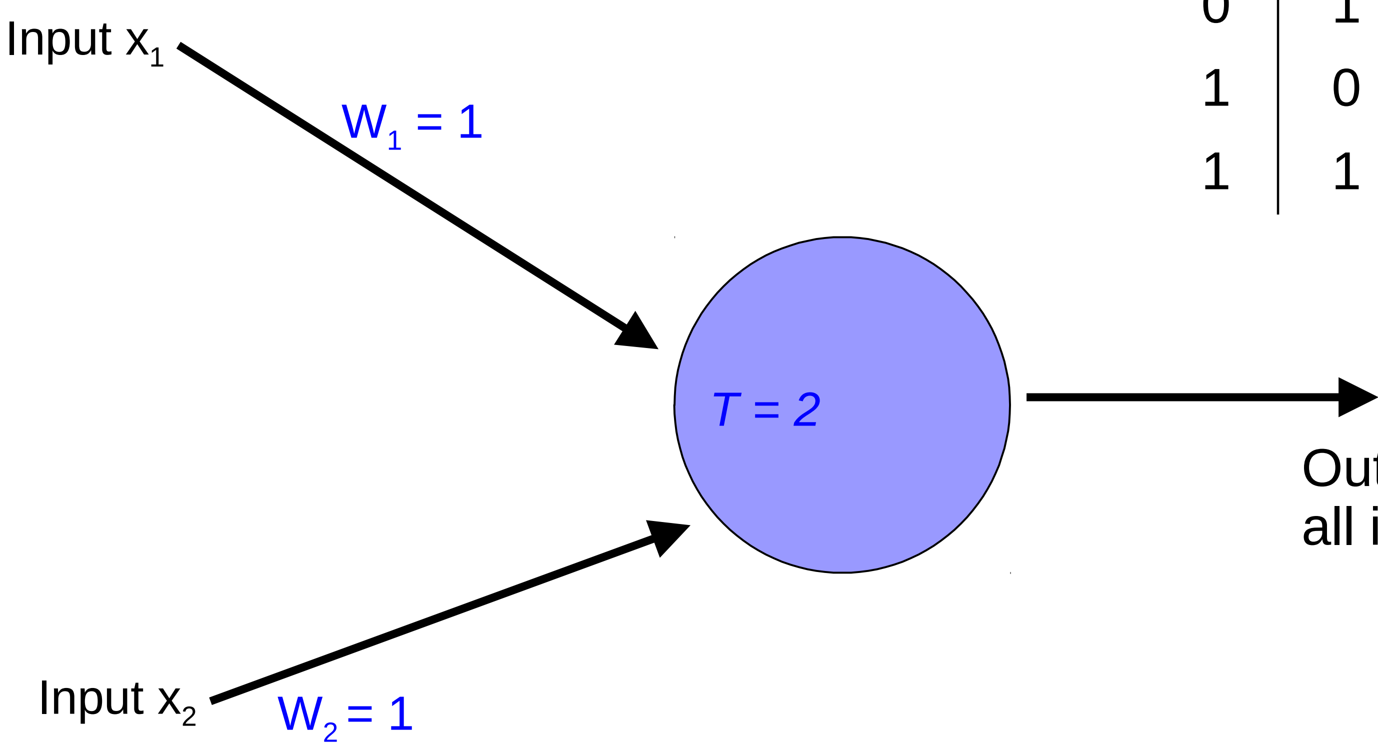
AND

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1



AND

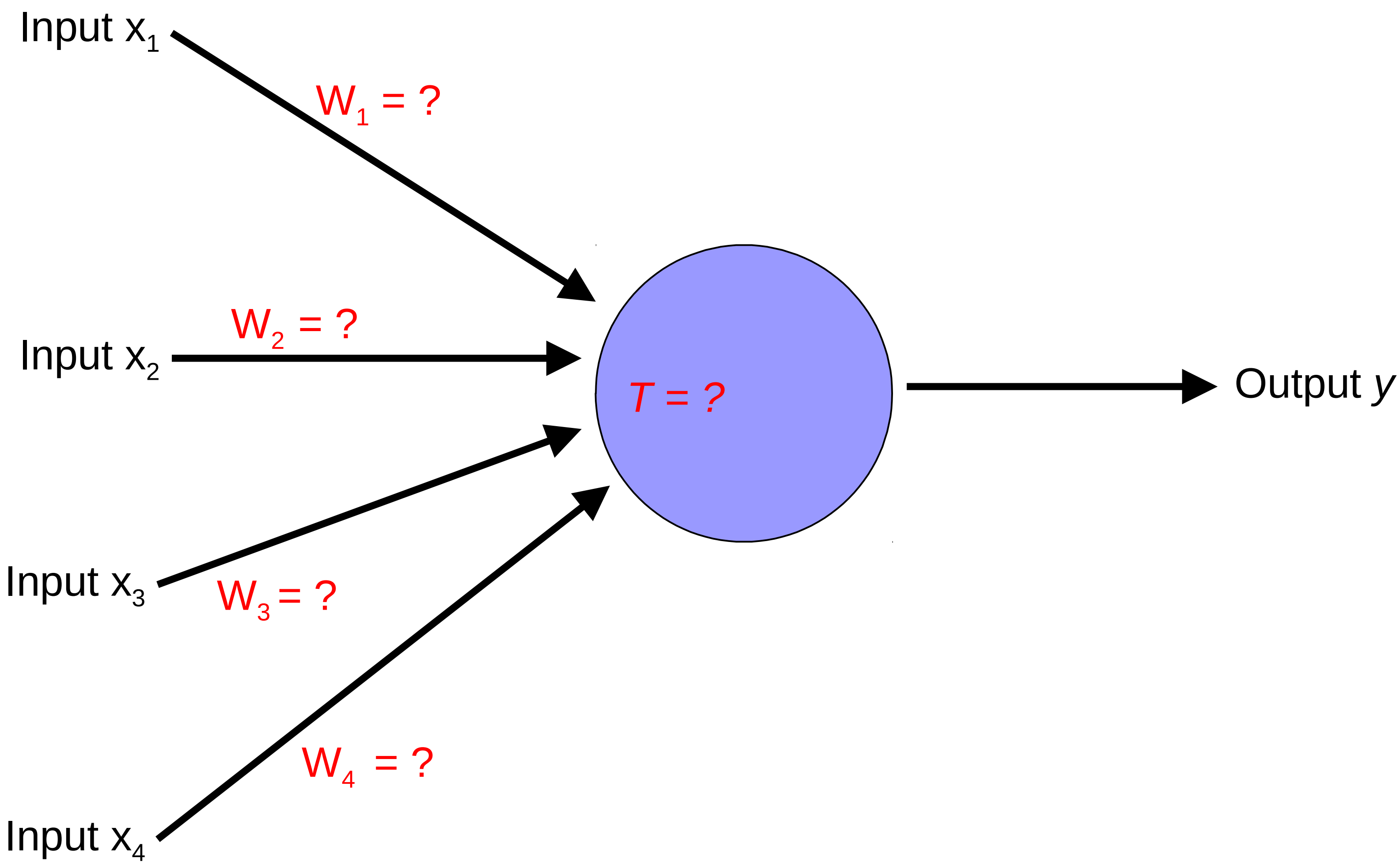
x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1



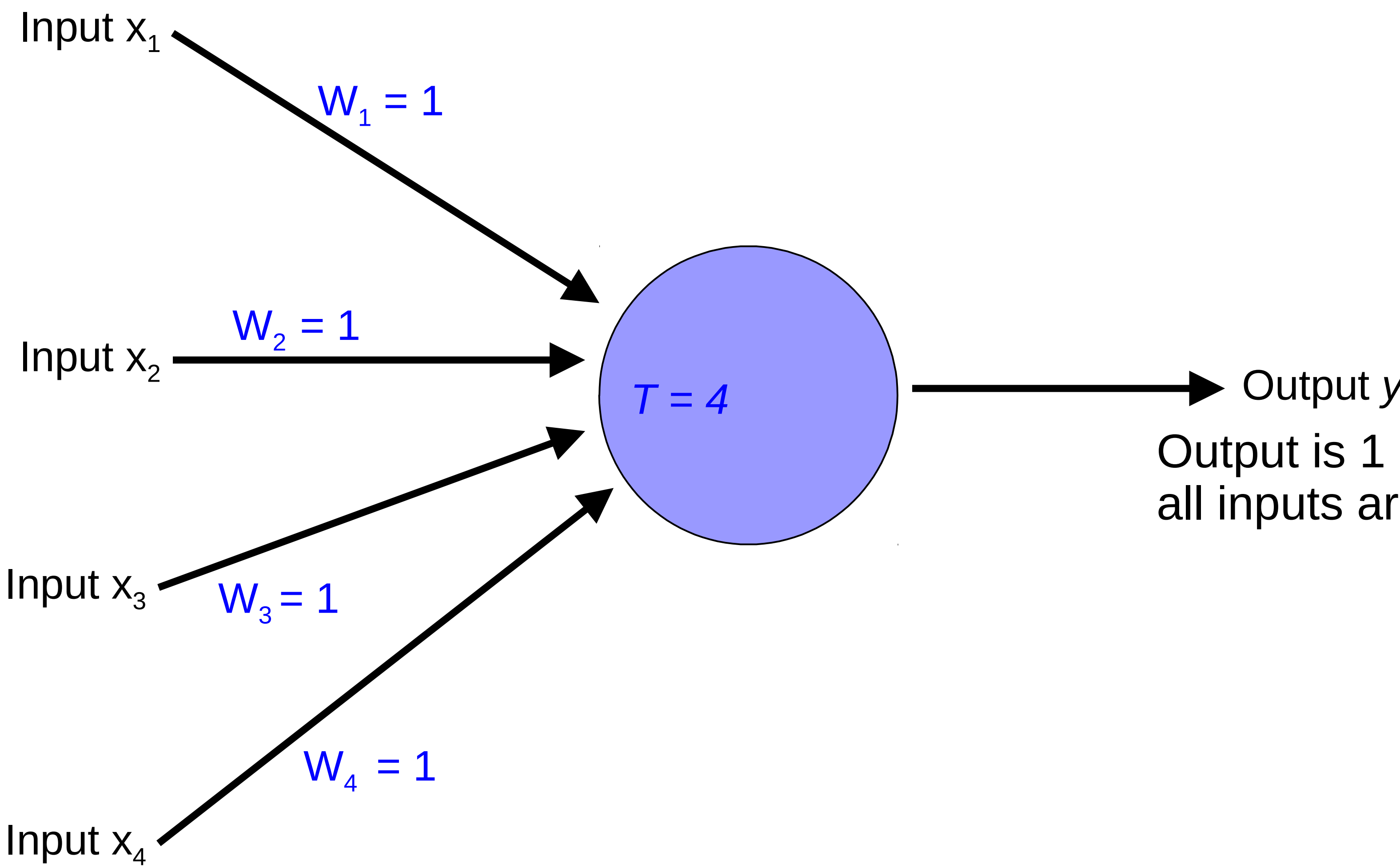
Output is 1 only if all inputs are 1

Inputs are either 0 or 1

AND



AND



Output is 1 only if all inputs are 1

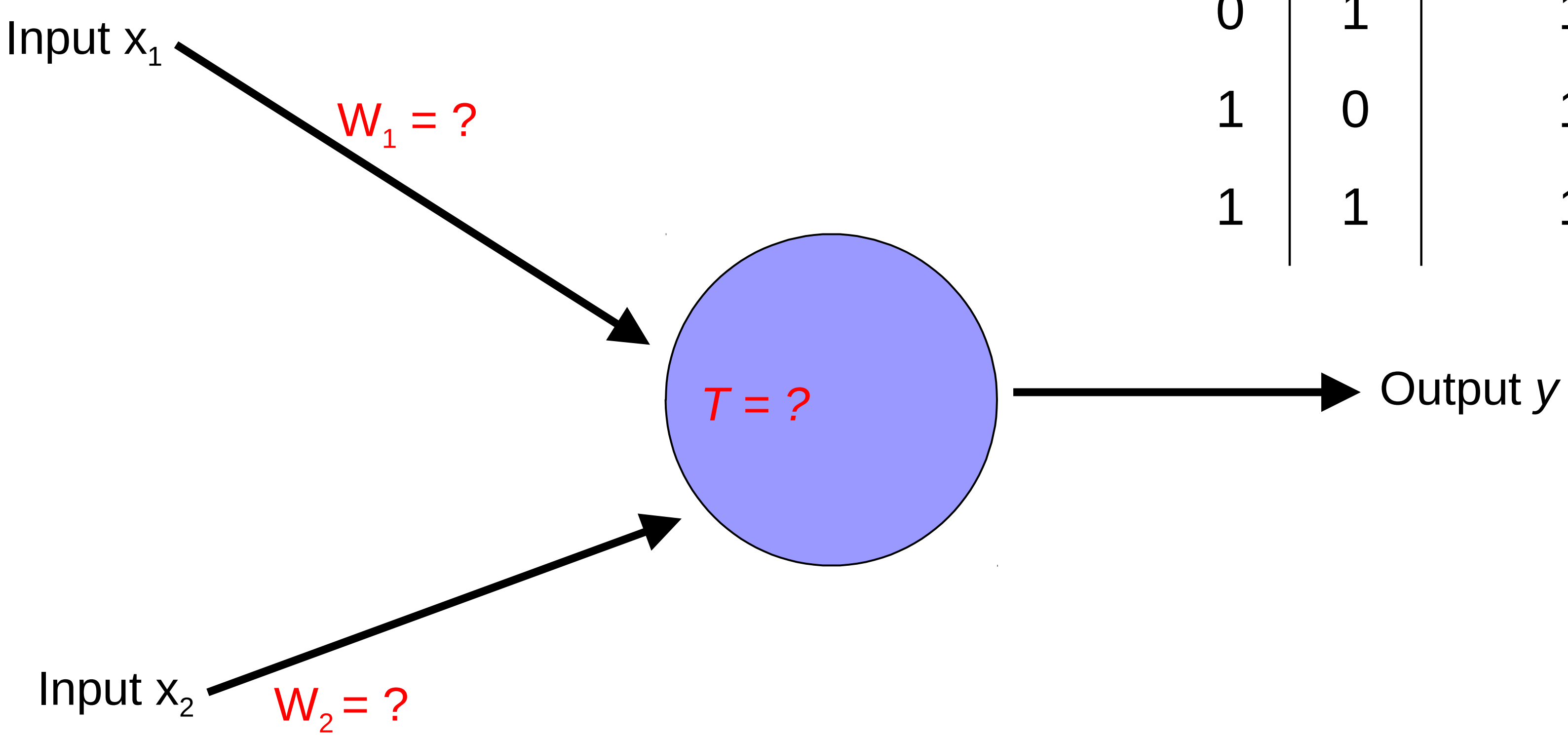
Inputs are either 0 or 1

OR

x_1	x_2	x_1 **or** x_2
0	0	0
0	1	1
1	0	1
1	1	1

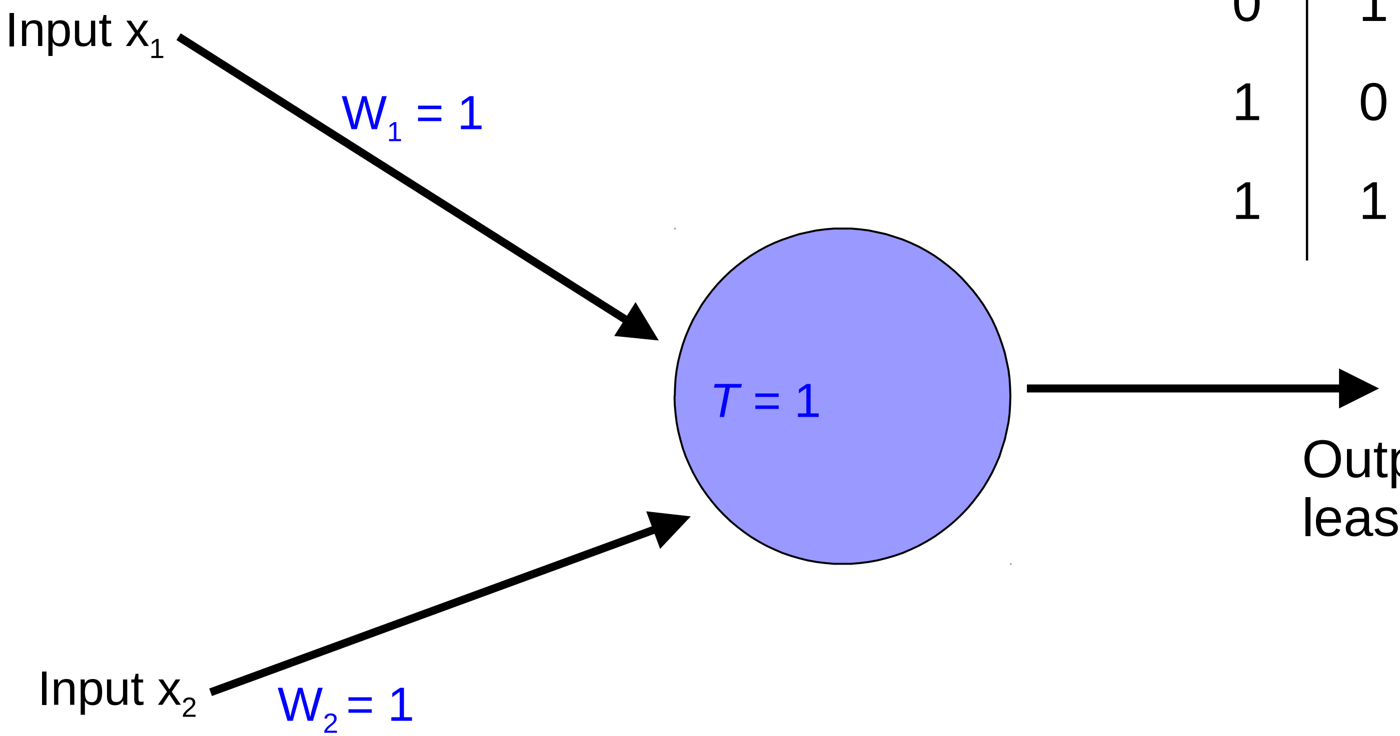
OR

x_1	x_2	x_1 **or** x_2
0	0	0
0	1	1
1	0	1
1	1	1



OR

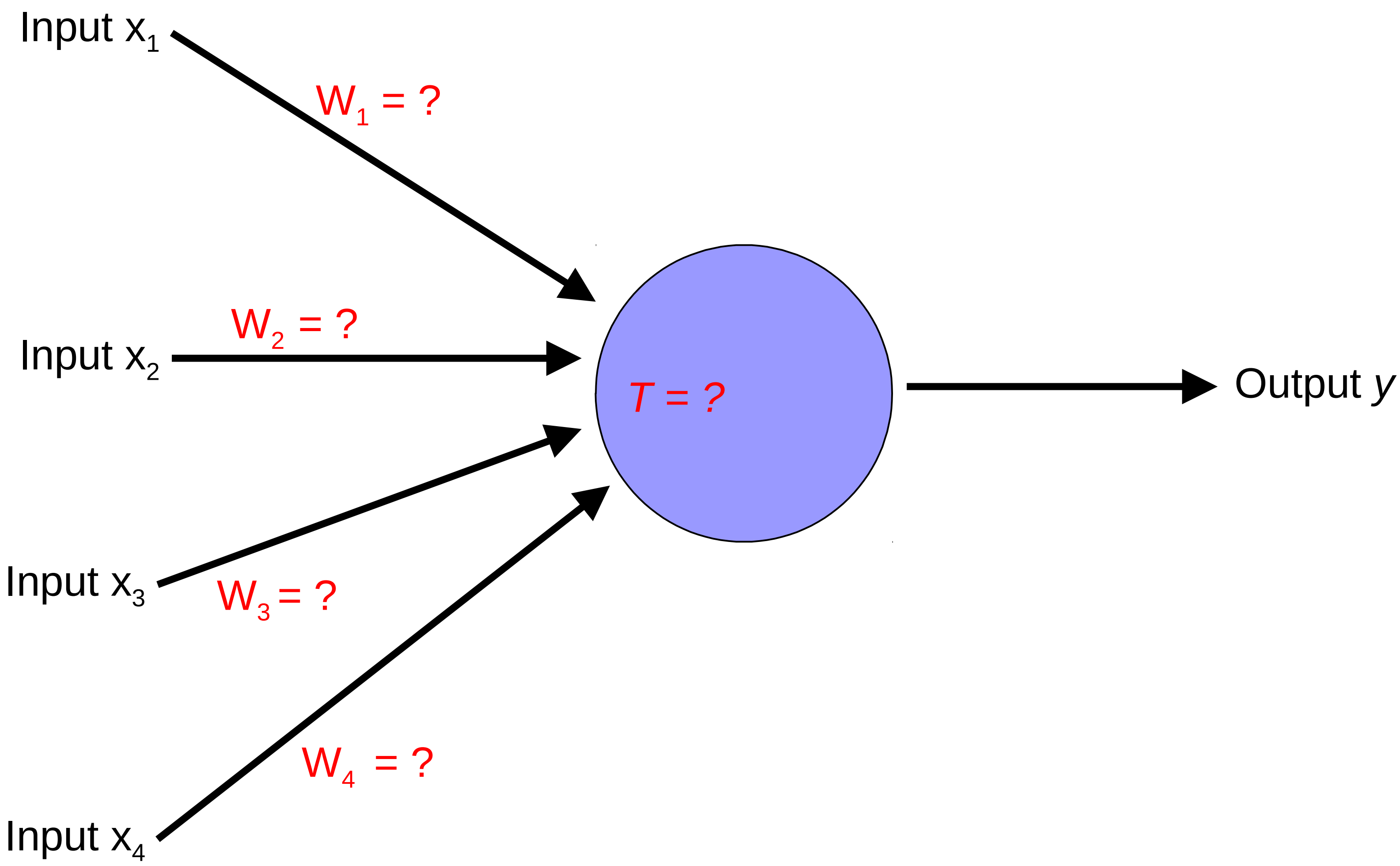
x_1	x_2	x_1 **or** x_2
0	0	0
0	1	1
1	0	1
1	1	1



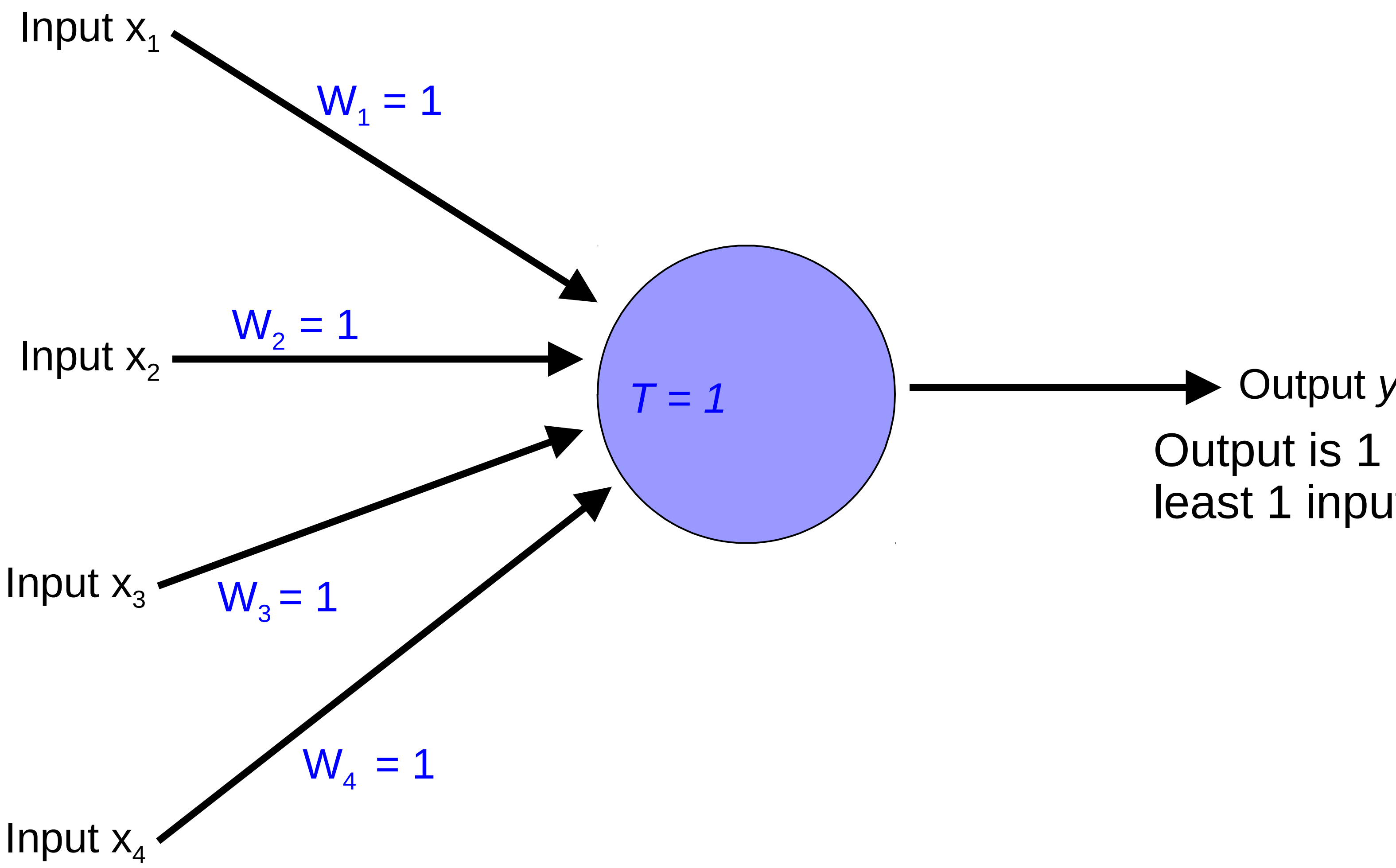
Output is 1 if at least 1 input is 1

Inputs are either 0 or 1

OR



OR



Output is 1 if at least 1 input is 1

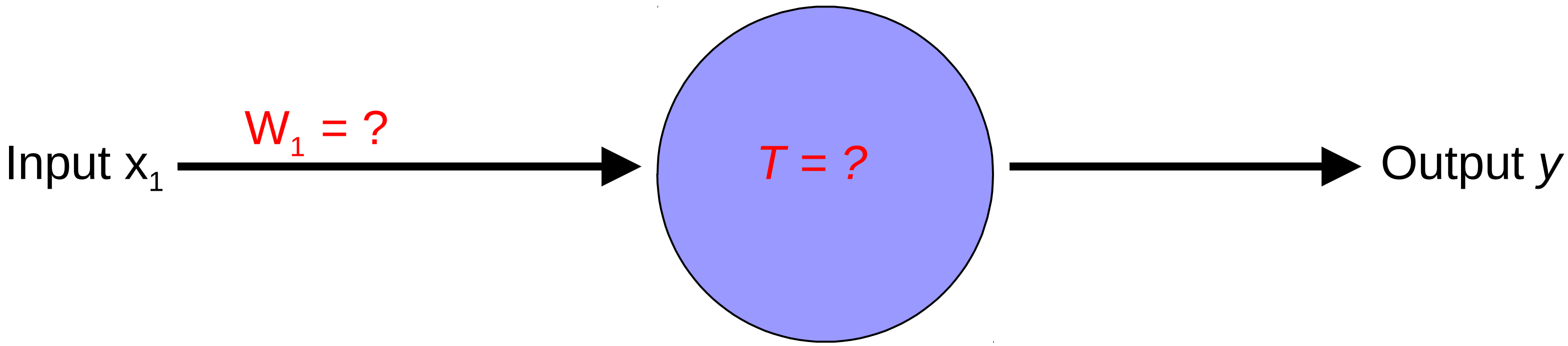
Inputs are either 0 or 1

NOT

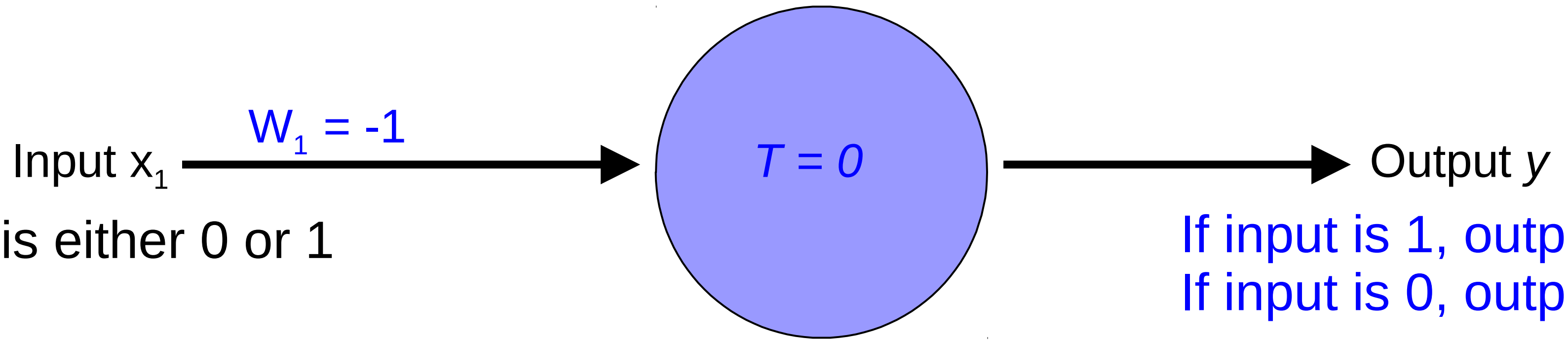
x_1	**not** x_1
0	1
1	0

NOT

x_1	**not** x_1
0	1
1	0



NOT

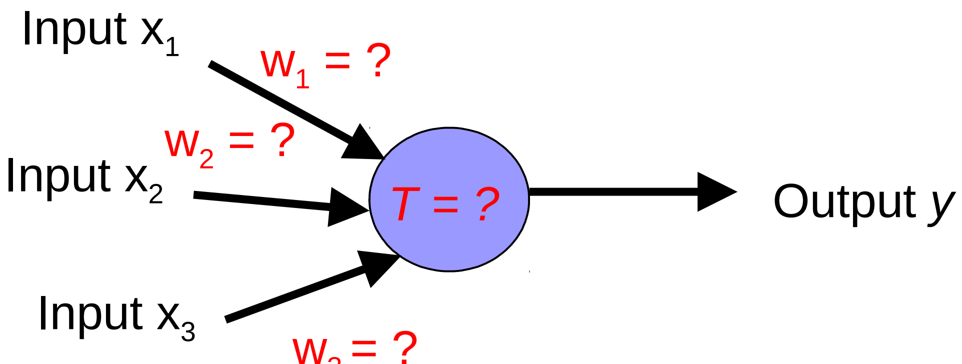


Input is either 0 or 1

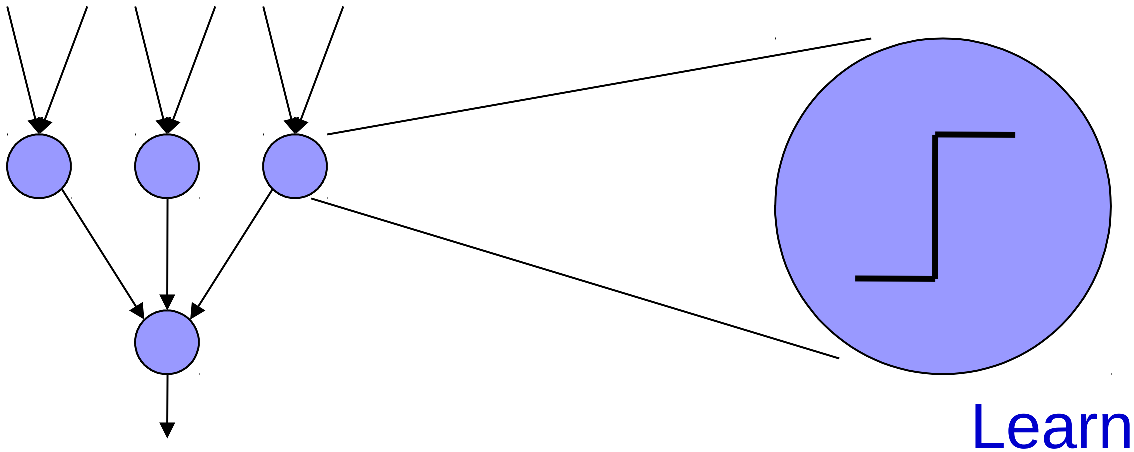
If input is 1, output is 0.
If input is 0, output is 1.

How about…

x_1	x_2	x_3	x_1 **op** x_2
0	0	0	1
0	1	0	0
1	0	0	1
1	1	0	0
0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	0



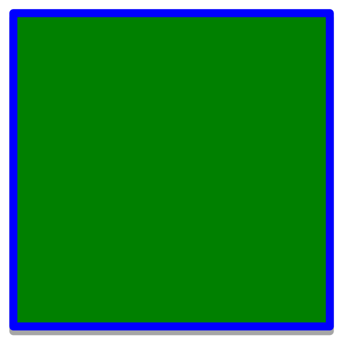
Training neural networks



Learn individual node parameters (e.g. threshold)

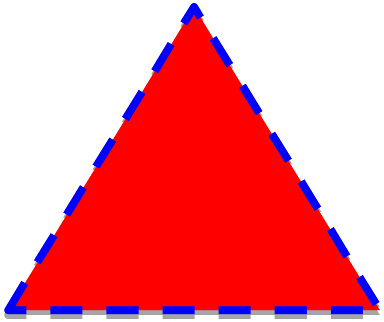
Learn the individual weights between nodes

Positive or negative?



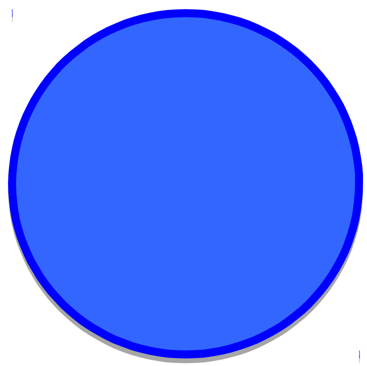
NEGATIVE

Positive or negative?



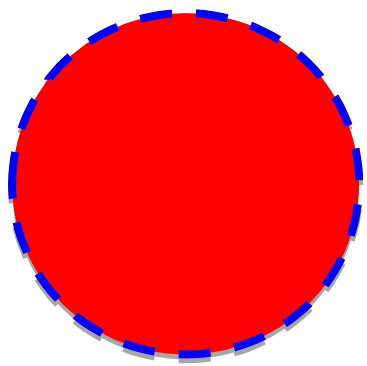
NEGATIVE

Positive or negative?



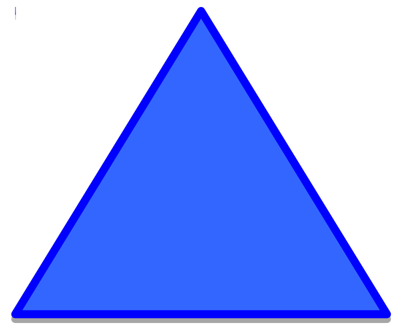
POSITIVE

Positive or negative?



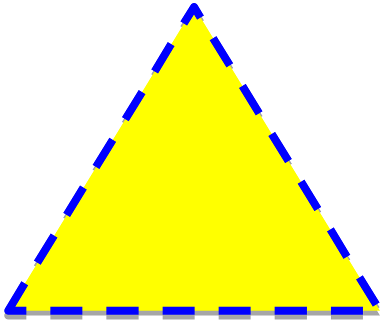
NEGATIVE

Positive or negative?



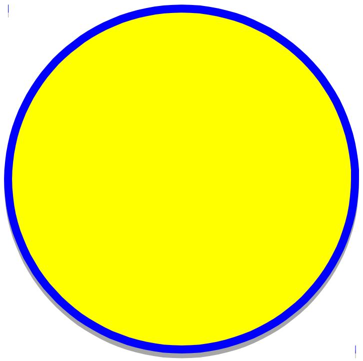
POSITIVE

Positive or negative?



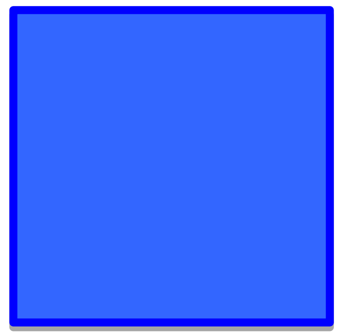
POSITIVE

Positive or negative?



NEGATIVE

Positive or negative?



POSITIVE

A method to the madness

blue = positive

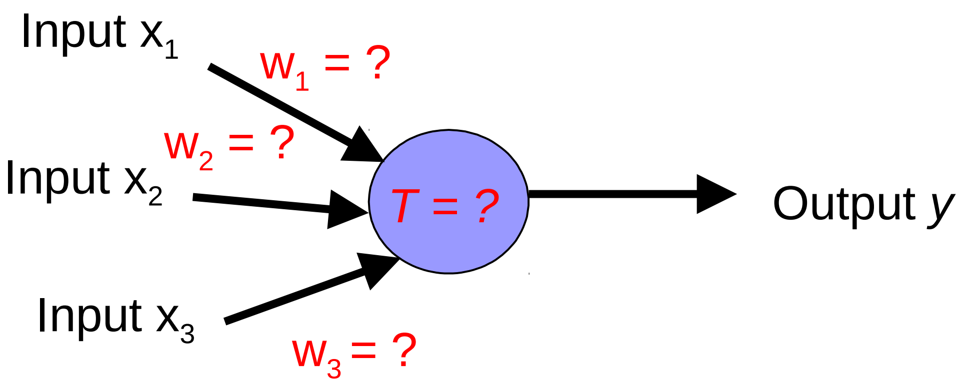
yellow triangles = positive

all others negative

How did you figure this out (or some of it)?

Training neural networks

x_1	x_2	x_3	x_1 **and** x_2
0	0	0	1
0	1	0	0
1	0	0	1
1	1	0	0
0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	0



1. start with some initial weights and thresholds
2. show examples repeatedly to NN
3. update weights/thresholds by comparing NN output to actual output